# SourceForge Automatic Release Packager for Ant

**Introduction**

The SourceForge File Publishing Tool provides a location to release bundled software, libraries, documentation, or other software development products. The File Publisher is organized into packages, releases, and files. One or more file constitutes a release and zero or more releases are bundled into packages. (Packages may be empty)

The SourceForge File Publisher Admin page is one way to create releases. The automatic release packager is another. The automatic release packager is an XML-RPC procedure that encapsulates the steps required.

**Ant**

The ant development tool (http://ant.apache.org) is a Java based build tool that uses XML definition files to specify tasks involved in building a particular software package. Common tasks include compiling source code, copying, moving and deleting files, and archiving compiled code into libraries or archives. The Ant tool is extensible with custom tasks.

**Custom Ant Task**

SourceForge provides a custom Ant task that can be used to create a new release in a SourceForge package directly from an Ant build. In this way, a single build process can compile, archive, package, and post a new release in SourceForge.

The actions necessary to do this are encapsulated in the provided Java class, SFPublishTask.class.

**Preparation**

In order to use the custom Ant task, the SFPublishTask class must be available to Ant. This is usually accomplished by putting the SFPublishTask.class file in the Java classpath, such as $JAVA_HOME/lib.

**Examples**

Once the SFPublishTask is available to Ant, the following line at the top of an Ant build file will define the task:

```
<taskdef name=”sf-publish” classname=”SFPublishTask” />
```

To use the packager, simply call the custom task with the appropriate arguments:

```
<sf-publish directory="release" server="sourceforge.example.com"
project="project_name" package="package_name" release="daily_04_10_2003"
  user="sf_user" password="sf_pass" maturity="Stable" />
```

All arguments are required, but can appear in any order:

| Argument Name | Notes |
|---|---|
| `directory` | The local directory where files to be released reside. All files in this directory will be added to the release. |
| `server` | The SourceForge server FQDN. |
| `project` | The short or unix name of the target project. |
| `package` | The package name where this release will be created. |
| `release` | The name of this release. If this release exists, and files of the same name exist, those files will be overwritten. If the release exists, files with filenames that do not match existing will be added to the release. |
| `user` | Username of a SourceForge server user that has RBAC priveleges to create releases. |

| | |
|---|---|
| password | The password of that user. |
| maturity | The maturity tag for the release. |

Complete Example of build.xml.

Blue Text indicates the custom task definition.
Red Text indicates a compile task.
Green Text indicates an archive task.
Orange text indicates the SourceForge automatic publish task. In this example, the product of the archive step (example.jar) is published to the SourceForge server in a release called "daily_04_10_2003."

```
<project name="example" default="basic" basedir=".">
        <description>
              Buildfile for Example SourceForge project
        </description>

        <taskdef name="sf-publish" classname="SFPublishTask"/>

        <property name="lib" location="lib" />
        <property name="src" location="src" />
        <property name="release" location="rel" />

        <target name="init">
                <tstamp/>
                <mkdir dir="${lib}" />
        </target>

        <target name="basic" depends="init">
                <javac srcdir="${src}" destdir="${lib}" />
        </target>

        <target name="clean">
                <echo>Cleaning up build products... </echo>
                <delete dir="${lib}"/>
        </target>

        <target name="archive" depends="basic">
                <jar jarfile="${release}/example.jar" basedir="${lib}" />
                <copy src="ssns.jar"
dest="/usr/local/jboss/server/default/deploy/" />
        </target>

        <target name="publish" depends="archive">
             <sf-publish directory="release" server="sourceforge.example.com"
project="project_name" package="package_name" release="daily_04_10_2003"
        user="sf_user" password="sf_pass" maturity="Stable" />
       </target>
</project>
```

Another example uses a separate XML properties file to store the username and password, so that a generic build.xml project can incorporate user specific credentials for the SourceForge software and avoids a plaintext password in a global build file. This example also uses the Ant "BuildNumber" task to create incrementally numbered releases.

<?xml version="1.0"?>

```
       <project name="SF-Publish Example" default="release" basedir=".">
              <taskdef name="sf-publish" classname="SFPublishTask"/>
              <target name="release">
```

```
                    <buildnumber />
                    <xmlproperty file="user.xml" />
                    <sf-publish directory="release"
server="sourceforge.example.com" project="project-name" package="package-name"
release="build-${build.number}" user="${user.name}" password="${user.password}"
maturity="Stable" />
             </target>
      </project>
```

The user.xml file contains:

```
      <user>
              <name>jdoe</name>
              <password>foo</password>
      </user>
```